Multithreading Workshop

Copyright © James Raynard 2021

Condition variable

- Rewrite the first condition variable example to use a boolean flag instead of a condition variable
- Your program must not contain any data race, deadlock or livelock conditions
- (The code is reproduced in the next two slides)

Copyright © James Raynard 2021

Condition Variable Example

```
// Global variables
condition_variable cv;                    // The condition variable instance
mutex mut;                                // The mutex used to protect the data
string sdata {"Empty"};                   // The shared data

// Waiting thread
void reader() {
   unique_lock<std::mutex> guard(mut);    // Acquire lock
   cv.wait(guard);                        // Unlock mutex and wait to be notified
   // Notification received               // Wake up and lock mutex
   cout << "Data is " << sdata << endl;   // Use the new value
}
```

Copyright © James Raynard 2021

Condition Variable Example

```
// Modyifing thread
void writer() {
    cout << "Writing data..." << endl;
    std::this_thread::sleep_for(2s);                    // Pretend to be busy...
    {
        lock_guard<std::mutex> lg(mut);                 // Acquire lock
        sdata = "Populated";                            // Modify the shared data
    }                                                   // Release the lock
    cv.notify_one();                                    // Notify the condition variable
}                                                       // Release the mutex
```

Copyright © James Raynard 2021

Condition Variables with Predicate

- Write a simple program with three threads
  - A fetch thread which fetches data over a network connection
  - A prog thread which displays a progress bar indicating how much data has been fetched
  - A processor thread which processes the data
  - (Do not attempt to implement the full functionality of these threads. Just provide print statements saying "fetching data" or similar)
- When all the data has been fetched, the progress bar is no longer displayed and the processor thread begins processing the data
- Make sure your program is not affected by data races, or spurious or lost wakeups

Copyright © James Raynard 2021

Loop

- Imagine the following function is executed concurrently by two threads

```
int x{0};

void func() {
    while (x == 0) {
        x = 1 - x;
    }
}
```

- Are there any possible scenarios in which neither thread is able to exit the loop? If so, how would you fix the problem?

Copyright © James Raynard 2021

Semaphore

- A semaphore is a synchronization object which maintains a count of available resources. It is often used when implementing pools (of threads, database connections, etc)
- The semaphore has a non-negative integral value, and two operations
  - put()
    - A resource is added or returned to the pool and the counter is incremented
  - take()
    - A resource is removed from the pool and the counter is decremented
- If the counter reaches zero, no more resources can be taken until the value becomes positive again
- Implement a basic semaphore type using a mutex and condition variable. Write a program to verify its behaviour

Copyright © James Raynard 2021